IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 09/660,005	§	Examiner:	Bengzon, Greg C.
Filed: September 12, 2000	§	Group/Art Unit:	2144
Inventors:	§	Atty. Dkt. No:	5181-66200
Thomas E. Saulpaugh	§		
Gregory L. Slaughter	§		
Michael J. Duigou	§		
	§		
Title: PRE-GENERATED MESSAGE ENDPOINTS	§		
	§		

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicants request review of the rejection in the above-identified application. No amendments are being filed with this request. This request is being filed with a notice of appeal. Claims 1-28 are pending in the application. Please note that for brevity, only the primary arguments directed to the independent claims are presented, and that additional arguments, e.g., directed to the subject matter of the dependent claims, will be presented if and when the case proceeds to Appeal.

The Examiner rejected claims 1-4, 8-17 and 21-28 under 35 U.S.C. § 102(e) as being anticipated by Weschler (U.S. Patent 6,842,903). Regarding claim 1, Weschler fails to disclose **receiving an address for a service within the distributed computing environment and linking the address to a pre-generated message interface for accessing the service**. The Examiner equates the address at which a plug-in module is stored with the address for a service of applicant's claim. However, the address at which a plug-in module is stored is not linked to a pre-generated message interface for accessing the service. Instead, Weschler teaches that the plug-in module may be stored locally or may not be stored on the same machine as core profile engine 201. In Weschler's system the plug-in module is loaded from the address at which it is stored and instantiated to provide additional functionality to core profile engine 201. Nowhere does Weschler mention linking the address at which a plug-in module is stored to a pre-generated message interface for accessing the service. Weschler describes, at the Examiner's cited passage (col: 8, lines: 5-10) that the plug-in modules are loaded from an address or fully qualified path pointing to a location at which the plug-in module is stored." Weschler does not disclose sending

messages to, or linking a pre-generated message interface to, the address from which a plug-in module was loaded. The Examiner has clearly overlooked this fact.

The Examiner's contention that since the plug-in module is associated with the address, Weschler disclosed *linking an address to a message interface* is clearly incorrect. The Examiner has improperly interpreted certain portions of Weschler. For example, col. 9, lines 20-25, cited by the Examiner, does not describe *generating an interface to the service*, as the Examiner suggests, but instead describes obtaining information about what interface the particular service version implements by issuing a getServiceInterface(version) method.

Additionally, the Examiner's reference to "casting to an interface" in Weschler is clearly not the same as linking an address to a pre-generated message interface for accessing the service, as recited in Applicants' claim. Casting from one programming type to another has absolutely nothing to do with linking an address to a pre-defined message interface for accessing a service. Weschler teaches converting the service interface (or a portion of it, or parameters to the interface) from one programming type (e.g., int, char, double, etc.) to another, which clearly has absolutely nothing to do with linking an address to a pre-defined message interface for accessing a service.

Regarding claim 8, Weschler fails to disclose **receiving a schema defining messages for accessing the service and generating message endpoint code according to the schema**. Weschler teaches a method for providing dynamic references between services in a computer system that allows one service to obtain a reference to another service without requiring specific knowledge of the other service and that Weschler teaches a service connector interface that encapsulates the logic necessary to locate an instance of a particular service.

The Examiner cites column 6, lines 25-35 and argues that "gaining a reference to data store adapters" discloses *receiving a schema defining messages for accessing the service*. The Examiner further submits that data store adapters would inherently involve a schema for accessing the data structures involved. However, even if Weschler's data store adapters involved a schema, there is nothing in Weschler that describes receiving a schema defining messages for accessing the data store, as recited in claim 8. Gaining a reference to a data store adaptor does not disclose, nor does it inherently include, receiving a schema that may, in the Examiner's opinion, be associated with the data store adaptor. Without some specific teaching by Weschler, the Examiner's contention that Weschler discloses *receiving a schema defining messages for accessing the service* is clearly improper.

Furthermore, the Examiner is incorrectly interpreting the general statements by Weschler that "the application casts to the interface" and that the "service connector may be compiled along with the application" as disclosing generating message endpoint code according to a schema. The general mention of casting to an interface and compiling a service connector "along with the application" do not in any manner disclose the specific limitation of generating code (i.e., *generating message endpoint code*) according to the schema, as recited in claim 8.

Applicants also note that the Examiner has never cited any portion of Weschler that discloses *loading the message endpoint code* (i.e., message endpoint code generated according to the message schema) *and operating code onto the device*, nor does Weschler teach this limitation of claim 8. **In fact, the Examiner merely states that Weschler discloses this limitation without providing any citation, interpretation, or explanation.**

The Examiner also rejected claims 1-4, 8-17 and 21-28 under 35 U.S.C. § 103(a) as being unpatentable over Roberts et al. (U.S. Patent 6,560,633) (hereinafter "Roberts") in view of Chen et al. (U.S. Publication 2002/0062334) (hereinafter "Chen"). Regarding claim 1, the combination of Roberts and Chen does not teach or suggest: **linking said address to a pre-generated message interface for accessing said service, wherein said pre-generated message interface is implemented by computer-executable code built in to said device during a code-build process for the device, wherein the pre-generated message interface is constructed prior to runtime**. Roberts' WSA interfaces are clearly meant to be downloaded and constructed *at runtime* and thus **teach away** from the use of computer-executable code built in to a device and comprising a pre-generated message interface for accessing a service, where the pre-generated message interface is constructed *prior* to runtime. See Applicants' previous response, pp. 10-13.

The Examiner also relies on Chen's built-in APIs. However, Chen's built-in APIs allows one of Chen's "actions" to create "a receiver thread and [to] register[] its socket address" (Chen, paragraph [0063]). APIs for creating threads and registering a socket address do not, even if combined with Roberts, teach or suggest a computer-executable code built into a device during a code-build process for the device and that comprises a pre-generated message interface for accessing the service.

The Examiner also contends that Roberts discloses a pre-generated message interface that was constructed prior to runtime, citing column 13, lines 15-20. However, the **Examiner is improperly**

ignoring the specific language recited in Applicants' claim. The Examiner has improperly ignored the fact that claim 1 recites a pre-generated message interface that is implemented by computer-executable code built in to said device during a code-build process for the device. Applicants' have shown that Roberts' WSA interfaces are clearly not built in to the devices on which they run prior to runtime. Instead, they are "child runtime models" dynamically constructed at runtime according to lists of features contained in templates (see, e.g., Roberts, column 10, lines 14-25).

Regarding claim 8, the Examiner has failed to provide a proper rejection under § 103 of this claim. The Examiner again lists claim 8 as rejected under 35 U.S.C. 103(a), but fails to provide an actual rejection. Claim 8 recites subject matter different from that recited by claim 1, the only claim for which the Examiner provides a rejection under 35 U.S.C. 103(a). Therefore, the rejection of claim 8 under § 103 is improper. Further regarding claim 8, Roberts in view of Chen fails to teach or suggest *pre-generating at least one message interface to be built-in to a device in order to access a service* by *receiving a schema defining messages for accessing the service; generating message endpoint code according to said schema; and linking said message endpoint code into executable operating code for the device and loading the message endpoint code and operating code onto the device.*

In the Response to Arguments section of the Final Action, the Examiner cites Roberts (column 7, lines 10-15 and lines 45-50) as teaching a regeneration process for a transformed runtime model and fully interactive user interfaces, and that the runtime models follow a schema. Applicants assert that this general reference to "a schema" does not teach the specific limitations of claim 8, which recite that message endpoint code (i.e., code for a pre-defined message interface) is generated according to a schema defining messages for accessing a service. Applicants assert that "a schema" for a runtime model may specify many different things and may or may not have anything to do with defining messages for accessing a service. **Nothing in Roberts teaches or suggests that this particular type of schema is used in generating the child runtime models**.

Roberts' regeneration process clearly does not teach pre-generating at least one message interface to be built-in to a device to access a service, as recited in claim 8. Instead, the regeneration process of Roberts is performed at runtime. Similarly, the user interfaces are described as "a set of dynamically generated and updated XML entities called Pages" in column 7, lines 11-13 (emphasis added.) Therefore, these child runtime models and user interfaces are clearly not pre-generated message interfaces that are generated according to the limitations of claim 8 and linked into executable operating code for a device. Furthermore, Roberts specifically teaches downloading WSA interfaces from web service devices for use

on client devices to access WSAs. Downloading of message interfaces is clearly quite different from linking message endpoint code, generated according to a schema defining messages for accessing a service, *into executable operating code* for a device and loading the message endpoint code and operating code onto the device. Thus, **Roberts teaches away** from Applicants' claim.

The Examiner also rejected claims 1-7, 12 and 25-28 under 35 U.S.C. § 112, second paragraph, as indefinite. Applicants traverse this rejection for at least the following reasons. The Examiner has improperly rejected individual phrases from Applicants' claim without considering the claim in its entirety and in light of Applicants' specification. For instance, the Examiner has overlooked Applicants' claim language regarding the pre-generated message interface being constructed prior to runtime and that it is implemented by computer-executable code built in to said device during a code-build process for the device. A code-build process is a well-understood and fundamental concept in the art of computer science and software/code application development. The Examiner is improperly considering the individual phrases of Applicants' claim, such as "built in" and "during code-build process" rather than considering the claim as read in its entirety and in light of Applicants' specification."

Regarding claims 12, 25, 26 and 28, Applicants again note that these claims do not include the phrase, "computer executable code built in to said device" and thus the rejection of these claims is improper. The Examiner has not provided any basis for the § 112, second paragraph, rejection of claims 12, 25, 26 and 28. Nor has the Examiner addressed this argument in the Response to Arguments.

If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicants hereby petition for such an extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 501505/5181-66200/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: April 26, 2007